PandaDB：一种面向异构数据的智能融合管理系统*

沈志宏[1],赵子豪[1,2],王华进[1],刘忠新[1],胡 川[1,2],周园春[1]

[1](中国科学院计算机网络信息中心 北京 100190)
[2](中国科学院大学 北京 100049)
通讯作者: 沈志宏, E-mail: bluejoe@cnic.cn

摘 要: 随着大数据应用的不断深入，大规模结构化、非结构化数据带来的异构数据的融合管理、关联计算和即席查询需求日益突出。现有异构数据融合管理技术与系统存在着数据模型表示能力弱、查询执行实时性差等问题。本文提出了适用于结构化、非结构化数据融合管理和语义计算的智能属性图模型，并定义了相关属性操作符和查询语法。基于该模型实现了异构数据融合管理系统 PandaDB，并详细介绍了 PandaDB 的总体架构、存储机制、查询机制、属性协存、AI 算法调度和分布式架构。测试实验和案例证明，PandaDB 的协存机制和分布式架构具备较好的性能加速效果，并可应用在关联数据发布、个人相册管理、学术图谱实体消歧等融合数据智能管理的场景。
关键词: 数据管理系统;融合管理;数据模型;数据查询;人工智能;
中图法分类号: TP311

PandaDB：An intelligent management system for heterogeneous data

SHEN Zhi-Hong[1], ZHAO Zi-Hao[1,2], WANG Hua-Jin[1], LIU Zhong-Xin[1], HU Chuan[1,2], ZHOU Yuan-Chun[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Due to the rapid development of big data applications, there is a increasing requirement of data fusion management, combined analysis and ad-hoc queries for large-scale structured / unstructured data. Existing heterogeneous data management technologies have some problems, such as weak data model representation ability and inefficient query execution. This paper proposes an intelligent property graph model suitable for structured and unstructured data fusion management and semantic computing, with some related property operators and query syntax. Based on this model, we implement a heterogeneous data fusion management system called PandaDB. This paper depicted the architecture, storage mechanism, query mechanism, property co-storage, AI algorithm scheduling and distributed architecture of PandaDB. Test experiments and cases show that the co-storage mechanism and distributed architecture of pandadb have good performance acceleration effects, and can be applied in some scenarios of fusion data intelligent management such as linked data publishing, personal photo album management, academic atlas entity disambiguation, and so on.

Key words: Data Management System; Data Fusion; Data Model; Data Query; AI;

1 引言

大数据时代，随着各类应用的推广使用，数据产生速度越来越快、数据体量越来越大。一方面，数据采集技术的迅猛发展使得数据的结构更多样、种类更加丰富，这种多元异构特点的一个重要体现是非结构化数据

* 基金项目: 国家自然科学基金重点项目 (61836013)，科技部创新方法工作专项(批准号: 2019IM020100)

Foundation item: Key Project of National Natural Science Foundation of China(61836013); Ministry of Science and Technology Innovation Methods Special work Project(2019IM020100)

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00

CNKI 在线出版时间: 0000-00-00

chinaXiv:202007.00035v1

占有较大比重。有研究表明，视频、音频、图片等非结构化数据占据高达 85%的比例[1]；另一方面，数据中台、知识图谱等数据管理/分析技术得到了广泛的应用。数据中台要求被管理数据能够同时支持多种应用；知识图谱要求对底层数据进行关联分析，并支持用户进行交互式查询，均提出了对结构化/非结构化数据进行融合管理、关联计算和即席查询的需求。

与结构化数据已具备完善成熟的数据库管理与分析系统相比，非结构化数据由于其结构和内容的复杂性，实现高效数据管理和分析具有较大的难度，具体体现为：1）管理难度大，相对于结构化数据，非结构化数据占有更大的空间，出于读写效率考虑，非结构化数据往往单独存在于文件系统，或者对象存储系统；2）分析难度大，非结构化数据内容比较复杂，传统查询技术只能针对元数据进行检索，无法查询非结构化数据所蕴含的丰富的内部信息；3）交互式查询，传统的 RDBMS 模型不适合对结构化数据和非结构化数据内在信息进行统一表示，因此在传统关系型数据模型下，无法实现对非结构化数据中信息的交互式查询。其根本原因有两方面：

一方面在于，传统的关系模型、属性图模型不能有效揭示和表达非结构化数据的内在信息。为了实现对非结构化数据内在信息的分析查询，需要从模型层面出发，设计相应的表达和查询方法。有学者提出将数据和 Schema 表示为边缘标记图，以此替代非结构化数据底层类型约束的缺失[2]。但该方法仅提出一种为非结构化数据添加 Schema 的方法，不能实现对非结构化数据中信息的自由检索。Li 等人提出从基本属性、语义特征、底层特征和原始数据等四个角度定义非结构化数据[3]，但这种方法依赖于预定义并不适用于非结构化数据的即时交互查询。近年来有学者提出在非结构化数据流上实施抽取 RDF 三元组的方法[4]，该方法只实现了三元组的抽取，不能支持对非结构化数据内部信息的交互式查询，且并不具备数据管理系统的基本能力。

另一方面在于，传统的数据管理方法是将结构化数据和非结构化数据分开存储的。非结构化数据通常存储在文件系统或者对象存储系统，并将其路径存储在关系数据库中；或者将非结构化数据在数据库中存储为二进制大对象（BLOB，Binary Large OBject），当应用获取数据的时候，返回一个二进制数组或者数据流。这两种方法在性能和功能上都不令人满意[1][5]。针对此问题，研究人员提出了一系列非结构化数据管理系统[3][6][7][8]，这些系统综合考虑了非结构化数据体积大、结构复杂的特点，设计了合适的存储模型，从底层存储出发，一定程度上解决了非结构化数据的存储和管理问题，但其提供的查询服务仅基于文件对象本身和元数据，不能提供对非结构化数据内部信息的查询能力。

综上所述，目前结构化/非结构化数据的融合管理存在如下难点：

- **传统的数据模型表示能力弱，无法实现异构数据中多元信息的统一表示**：非结构化数据内在信息属于原始数据中的衍生信息，这种信息在针对数据本身的建模方法中缺少合理的表示。应设计合理的数据模型，实现对异构数据内在多元信息的统一表达；
- **对异构数据进行关联分析的实时性差，无法支持交互式查询**：系统应能动态地响应查询请求。当前非结构化数据信息的自动抽取依赖于 AI 算法。因此应该设计合理的 AI 算法集成机制，使系统支持对非结构化数据信息抽取能力的动态扩展；

为实现对结构化/非结构化数据的融合管理、关联计算和即席查询，本文提出了智能属性图模型及其查询语法。智能属性图模型在属性图的基础增加了对非结构化数据的表达能力，以及结构化和非结构化数据之间的互操作能力。

本文在第 2 节给出了属性图扩展模型和相关概念，包括层叠属性图、智能属性图、次级属性等，并提出了属性操作符和查询语法。在第 3 节中给出了基于智能属性图模型的一个系统设计和具体实现。在第 4 节中通过实验和案例验证了该系统的效率及可行性。第 5 节介绍了与本文研究相关的工作。最后，对未来研究可能面临的挑战进行了展望。

2 概念设计

传统属性图模型无法有效表达非结构化属性，本节提出属性图扩展模型，以解决非结构化属性的有效表

chinaXiv:202007.00035v1

达问题，然后介绍针对属性图扩展模型的语义操作和查询语法设计，以支持因引入非结构化属性及其次级属性带来的新查询特性。

2.1 属性图扩展模型

传统属性图模型可以形式化表示为 G=(V, E, P)，其中 G 表示全体数据，V 表示数据中的实体集合，E 表示实体间的关系集合，P 表示数据集中实体的属性集合。

针对图片、语音、文本这样的非结构化属性，属性图模型无法有效揭示其蕴含的内在信息（如：某类顶点的 photo 属性蕴含有“车牌号”信息），内在信息的表达通常是离线且不完全的：

- **离线**：将非结构化属性转化为结构化、半结构化属性的 ETL 过程属于对数据的预处理；
- **不完全**：ETL 方式需要根据所需提取的目标结构化、半结构化属性逐案实现，无法枚举其中蕴含的全部结构化、非结构化属性。

为增强对非结构化属性的描述能力，本文针对属性图模型进行了扩展，提出层叠属性图模型和智能属性图模型。

定义 2-1：具备以下特征的属性图被称为层叠属性图(Cascading Property Graph)：

1) 层叠属性图 G^c 可以表示为 G^c=（V, E, P^P, P^N），其中 P^P 是基本属性（Primitive Property）集合，P^N 是内嵌式属性（Nested Property）集合；
2) 基本属性的值为文本、数值等基本数据类型；
3) 内嵌式属性的值为另外一个属性图；

定义 2-2：具备以下特征的属性图被称为智能属性图（Intelligent Property Graph）：

1) 智能属性图 G^I 可以表示为 G^I=（V, E, P^I），其中 P^I 为智能属性;
2) 智能属性 P^I 具有 P^P/P^N 二象性，即具有基本属性和内嵌式属性两个状态；
3) 在 G^I 中，存在展开操作ψ，可满足 $P^N=\psi(P^P)$，即将基本属性 P^P 展开为内嵌式属性 P^N；
4) 在 G^I 中，存在语义计算操作ξ，可满足 $\sigma=\xi(G^I_1,G^I_2)$，即针对 G^I_1 和 G^I_2 两个智能属性进行语义计算，得到结果σ；

定义 2-3：针对内嵌式属性 P^N_i，其具有次级属性（Sub-property）P^{iS}:

1) 存在 $V_i, P^N_i \in G^c$，V_i 是层叠属性图中的节点，P^N_i 是某个节点 V_i 的内嵌式属性；
2) P^N_i 的值可以表示为 G_{ni}，$G_{ni} - V_n = Ø$; $V_n - V_{n1} = Ø$ P_{ni} 的值可以表示为属性图 G_{ni}，且 G_{ni} 的顶点集 V_n 中仅包含一个顶点 V_{n1};
3) 顶点 V_{n1} 的属性集合 P^{1S}，其中的任一元素 P^{1S}_j 被称为 V_i 的次级属性；

采用智能属性，可以表达结构化属性和非结构化属性。图 1 示出一个智能属性图，针对 Car 类型的顶点 car1，具有一个内嵌式属性 photo，执行展开操作后，car1 具有两个次级属性：photo->plateNumber 和 photo->model。

chinaXiv:202007.00035v1

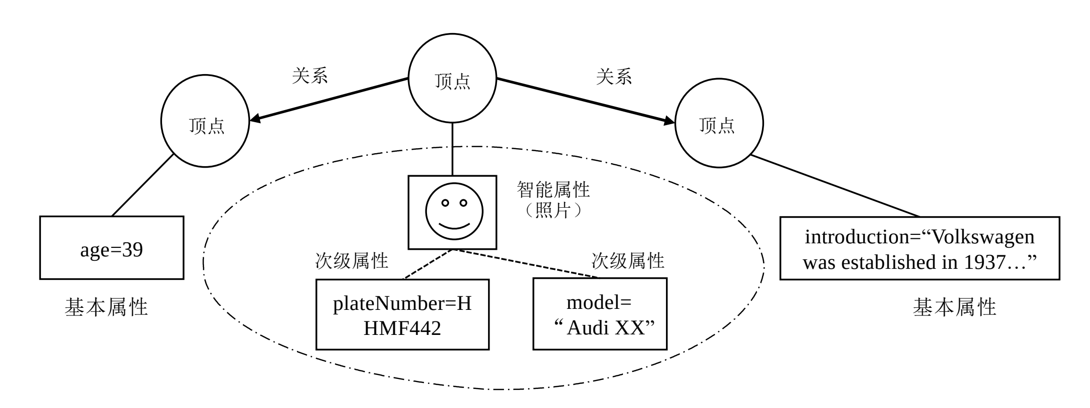


Fig.1　Intelligent Property Graph Model

图 1　智能属性图模型

chinaXiv:202007.00035v1

2.2 属性操作符

根据定义 2-2，本文定义了针对智能属性的次级属性抽取操作符和语义相似度操作符：

- **次级属性抽取操作符：**次级属性抽取操作符->用以从抽取智能属性中蕴含的次级属性，如针对 photo 属性执行 photo->plateNumber，即可抽取到 photo 中的车牌号；
- **语义相似度操作符：**传统属性图查询语言中的谓词（Predicate），只支持对属性进行比较，如=、>、<、正则匹配等。本文针对属性新增~:、::、:>等拓展谓词，以表达非结构化属性之间的相似关系、相似度、包含等关系。

Table.1　Semantic Similarity Operators

表 1　语义相似度操作符

操作名称	符号	含义	示例
SemanticCompare	::	计算 x 和 y 之间的相似度	x::y=0.7
SemanticLike	~:	计算 x 和 y 是否相似？	x~:y=true
SemanticUnlike	!:	计算 x 和 y 是否不相似？	x!:y=false
SemanticIn	<:	计算 x 是否在 y 里	x<:y=true
SemanticContain	>:	计算 x 是否包含 y	y>:x=true

在属性图模型中，非结构化属性由于其内在信息的不可见性，非结构化属性之间的计算操作支持有限。根据定义 2-1、2-2、2-3 和属性操作符的特性，智能属性图模型可以实现非结构化属性内在信息的在线、完全表达。

- **在线：**从非结构化属性中提取结构化、半结构化信息（即次级属性）的过程是按需触发的，无需对非结构化属性进行专门的预处理。
- **完全：** 在底层查询机制的支持下，可以实现针对非结构化属性的直接运算；次级属性根据查询语句生成，无需预先定义，因而也无需逐案实现从非结构化属性中提取各种蕴含属性的 ETL 过程。

因此，智能属性图模型非常适合结构化/非结构化数据的统一表达，基于智能属性图模型构建的数据库管理系统，可实现结构化/非结构化数据的融合管理。

2.3 查询语法

本文针对智能属性图模型，基于标准化的 Cypher 查询语言进行扩展，形成 CypherPlus 语言。CypherPlus

语言在语法层面引入次级属性抽取操作符、非结构化属性操作符、非结构化属性字面值等新特性，从而支持对非结构化属性的表达和语义操作。

（1）BlobLiteral：用于表达非结构化属性字面值，格式如<schema://path>，其中 schema 可以为 FILE、HTTP(S)、FTP(S)、BASE64 等多种类型。如图 2 所示；

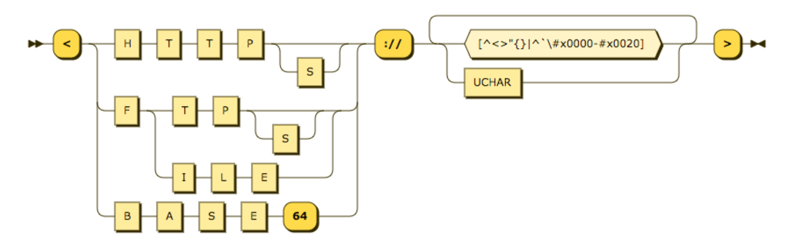


Fig.2 Grammer definition of BlobLiteral

图 2 BlobLiteral 语法定义

（2）SubPropertyExtraction：用于表达次级属性的抽取操作，如图 3 所示，其中 PropertyKeyName 为次级属性的名称；

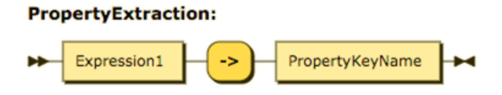


Fig.3 Grammer definition of SubPropertyExtration

图 3 SubPropertyExtraction 语法定义

（3）SemanticComparison：包括 SemanticCompare、SemanticLike、SemanticUnlike、SemanticIn、SemanticContain 等操作符。以 SemanticLike 为例，它用以指示两个属性的值是否相似，语法定义如图 4 所示，其中 AlgorithmName 为指定计算的算法名称，Threshold 为阈值，AlgorithmName 和 Threshold 为可选项，该情况下执行引擎则采用默认的比较器和阈值。

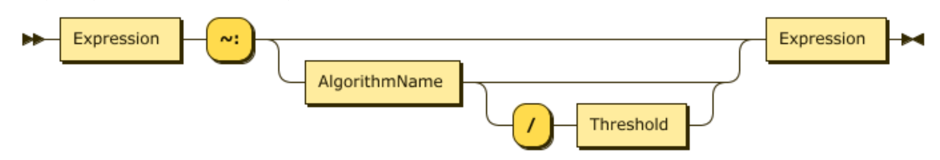


Fig.4 Grammer definition of SemanticLike

图 4 SemanticLike 语法定义

例如，针对图 1 的数据模型，可以查找与车牌号为’HHMF442’的车相似的车，查询语句如下：

Q1: match (c1:CAR), (c2:CAR) where c1.photo~:c2.photo and c1->plateNumber=’HHMF442’ return c2;

查询语句 Q1 可以形式化地描述为：

$$\pi_B(:\sim_{A,B_{phtoto}} \cap F_{\mathrm{A}_{photo \rightarrow plateNumber}} \cap F_{A,B_{label:Car}})$$

3 系统实现

为实现结构化、非结构化数据的融合管理和关联查询分析，本文采用智能属性图模型，基于 Neo4j 开源版本，设计并实现了原生支持 AI 图数据库管理系统 PandaDB。本部分 3.1 小节介绍 PandaDB 的总体架构，3.2 到 3.6 分别介绍各模块的设计思路和实现细节。

chinaXiv:202007.00035v1

3.1 总体架构

PandaDB 以智能属性图模型的形式组织数据，数据的存储形态被分为图结构数据、结构化属性数据和非结构化属性数据三部分。其中，图结构数据指图的节点和边等描述图结构的数据；结构化属性数据指数值、字符串、日期等类型的数据；非结构化属性数据泛指除结构化数据之外的数据，如视频、音频、图片、文档等。PandaDB 以 BLOB 对象的形式存储非结构化数据，并将其表示为实体（节点）的属性。根据上述三类数据的应用特点，PandaDB 设计了分布式多元存储方案：

- **分布式图数据**：基于传统的图数据库保存图结构数据和属性数据，在每个节点上保存相同的数据副本；
- **结构化属性协存**：基于 ElasticSearch、Solr 等外部存储实现大规模结构化属性数据的索引；
- **BLOB 存储**：基于 HBase、Ceph 等存储系统实现非结构化属性数据的分布式存储；

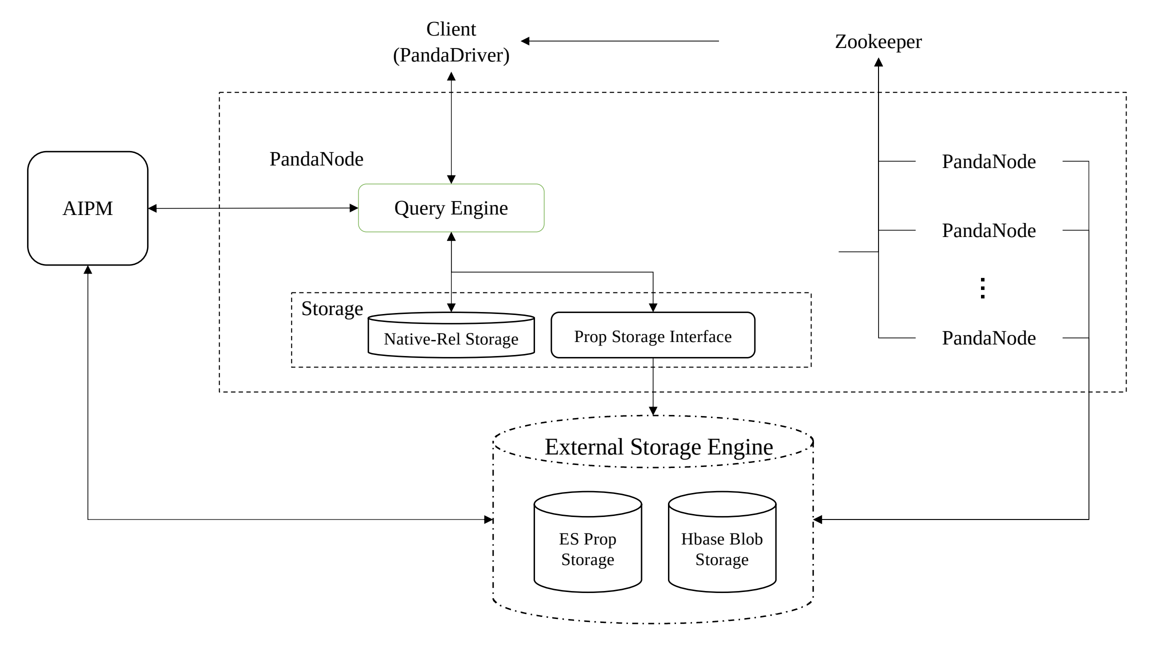


Fig.5 Architecture of PandaDB

图 5 PandaDB 总体架构设计

PandaDB 总体架构如图 5 所示，重要模块描述如下：

- **存储引擎**：维护本地的图结构数据，调度外部属性存储，按需为查询引擎提供服务；
- **外部存储**：包括基于 ElasticSearch 的结构化属性协存和基于 HBase 的 BLOB 存储两部分；
- **查询引擎**：解析并执行 CypherPlus 查询；
- **AIPM**：AI 模型服务框架，通过模型和资源管理，实现 AI 模型的灵活部署、高效按需运行，同时有效屏蔽 AI 模型之间的依赖。

PandaDB 集群采用了无主架构设计，其中 PandaNode 是单个节点，包含查询引擎和存储引擎两部分。属性数据和非结构化数据存储在外置分布式存储工具中，单个节点只保存图结构数据，通过属性存储接口与外置存储交互。

3.2 存储机制

PandaDB 将 BLOB 引入了 Neo4j 的类型系统，同时对 Neo4j 的存储结构进行了改造。存储结构如图 6 所示，除了 Neo4j 使用区，BLOB 的属性字段同时记录了 BLOB 的元数据，包括：唯一标识 blobid、长度 length

chinaXiv:202007.00035v1

和 MIME 类型。

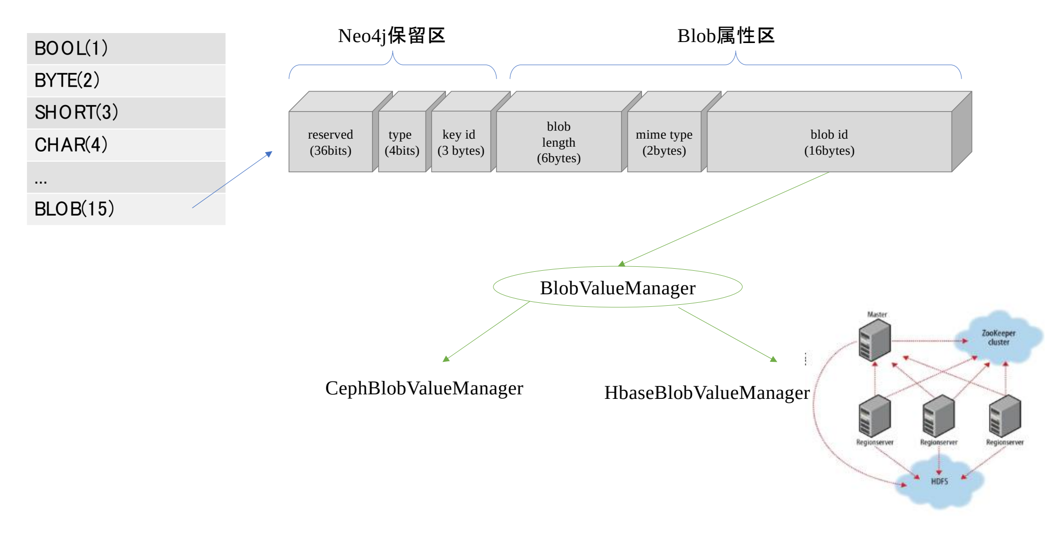


Fig.6 Design of BLOB storage structure

图 6 BLOB 存储结构设计

为实现对外部 BLOB 存储系统的调用，设计 BlobValueManager 接口，定义了 getById()/store()/discard()等操作方法。作为 BlobValueManager 的一个实现，HbaseBlobValueManager 基于 Hbase 集群实现 BLOB 数据的存取。在该方案中，为支持大规模 BLOB 的存储，Hbase 被设计成包含 100 个列的宽表，采用 blobid/100 作为 HBase 表的 rowkey，blob%100 对应于 Hbase 的某一列。

为加速 BLOB 的读取，BLOB 的内容读取被封装为一个 InputStream，在用户通过 Bolt 协议获取 BLOB 内容或进行语义计算的时候，这种流式读取的机制提高了运行的性能。

多元存储带来了存储事务安全的复杂性，客户端在写入数据时，将写操作请求发送到 PandaDB 的 Leader 节点，然后由 Leader 节点执行具体的写入操作。如图 7 所示，Leader 节点的具体操作流程如下：

（1）Leader 节点开启事务，执行 Cypher 解析，翻译成具体的执行操作；

（2）向 BLOB 存储引擎发送请求，执行 BLOB 数据的写入操作。若执行失败，则向上回滚，标记事务失败；

（3）若 BLOB 数据写入成功，则执行图结构数据和结构化属性数据的写入操作。若执行失败，则向上回滚，标记事务失败；

（4）将结构化属性数据的修改，同步到协存。若执行失败，则向上回滚，标记事务失败；

（5）执行事务提交；

（6）关闭事务，返回操作成功。

chinaXiv:202007.00035v1

chinaXiv:202007.00035v1

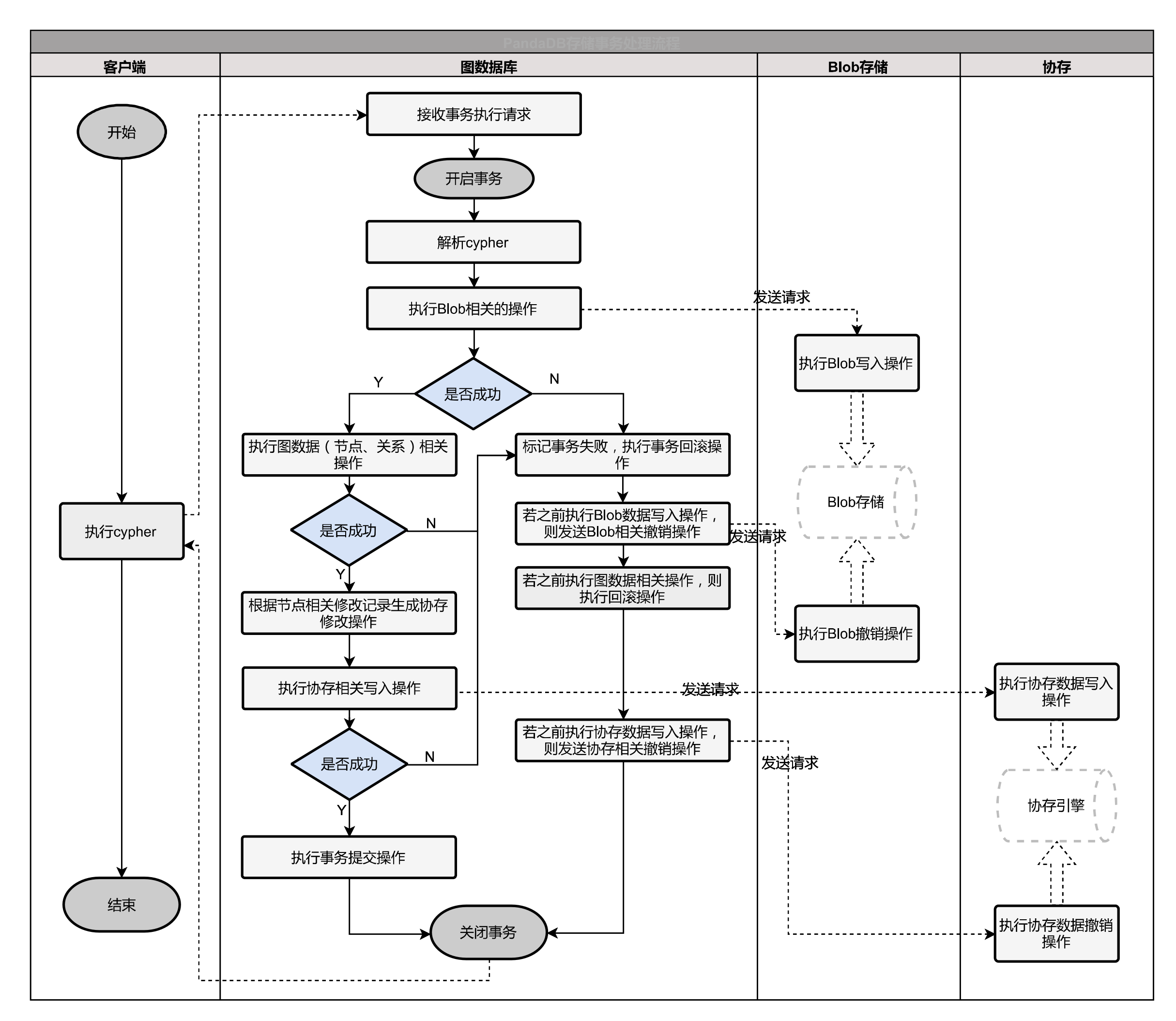


Fig.7 Write Transaction process in PandaDB

图 7 PandaDB 存储事务处理流程

3.3 查询机制

PandaDB 查询引擎主要实现查询语句的解析、逻辑计划的生成与优化、物理计划的选择与执行。基于 Neo4j，PandaDB 查询引擎主要改进如下几个部分：

（1）解析阶段：增强 Cypher 语言的解析规则，支持 BLOB 字面常量（BlobLiteral）、BLOB 二级属性的抽取操作符（SubPropertyExtraction），以及属性语义计算操作符（SemanticComparison）；

（2）语法检查阶段：针对 BlobLiteral、SubPropertyExtraction、SemanticComparison 执行形式检查，如发现非法的 BLOB 路径，非法的语义算子和阈值等；

（3）计划优化阶段：针对 BlobLiteral 的操作进行优化，针对大规模属性过滤情形，采用谓词下推策略等；

（4）计划执行阶段：充分调度属性协存模块、AIPM 模块，以及 BLOB 存储模块，实现高效的属性优先过滤、BLOB 获取与语义计算；一个典型的查询流程如图 8 所示。

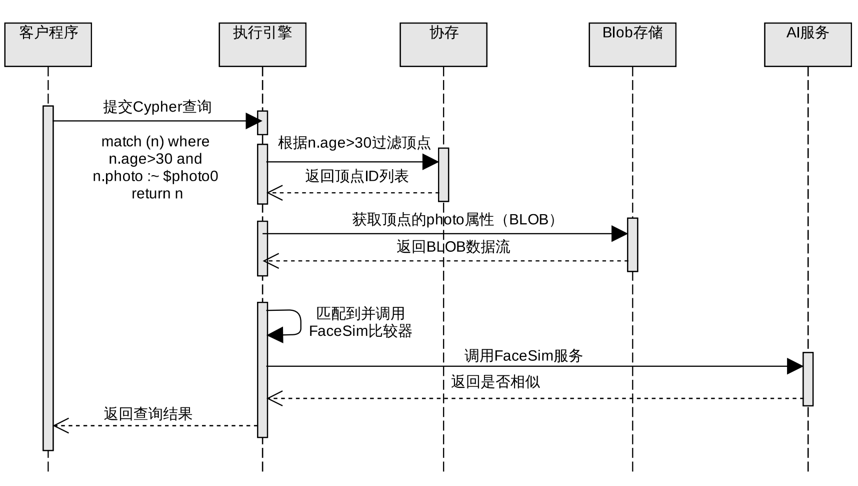


Fig.8 CypherPlus Query Process

图 8 CypherPlus 查询流程

图 9 从查询语法、查询计划、执行引擎三个层面示出 PandaDB 查询机制的设计。

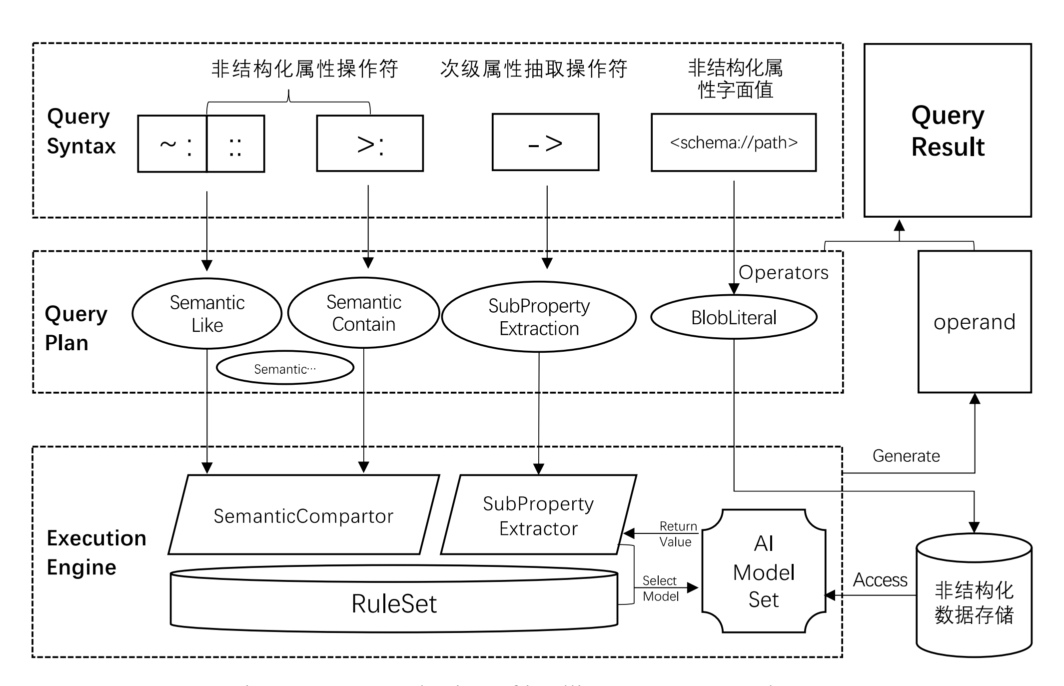


Fig.9 Query Mechanism of intelligent property graph

图 9 智能属性图查询机制

3.4 属性数据协存

PandaDB 引入属性数据协存机制，主要加速节点属性的过滤查询效率、实现属性数据的全文索引。目前，PandaDB 支持 ElasticSearch 作为协存引擎。

chinaXiv:202007.00035v1

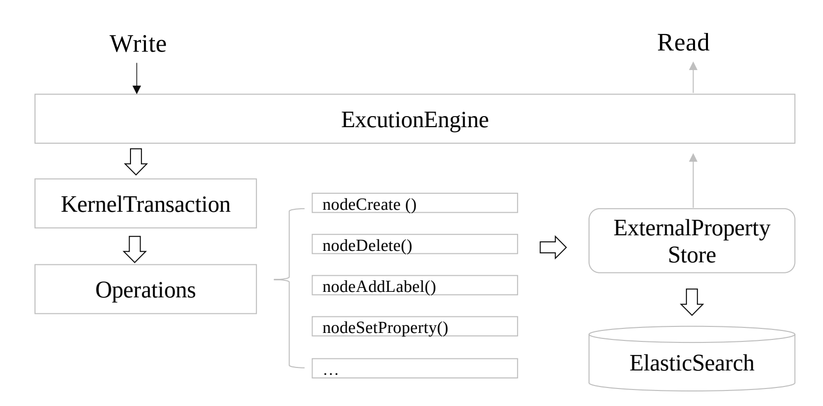


Fig.10　Write progress of property co-store module in PandaDB

图 10　PandaDB 协存模块的写入流程

图 10 示出 PandaDB 协存模块的写入流程，实现方法描述如下：

（1）**属性数据在 ElasticSearch 中的存储结构**：每个 Neo4j 图数据库对应协存引擎（ElasticSearch）中的一个独立的索引，每个节点的属性数据和标签数据均组织成 ElasticSearch 中的一个文档，其中节点在 Neo4j 数据库中的 ID 作为文档的 ID，节点属性标签作为文档的属性标签，节点属性数据作为文档的属性数据，设置特殊的属性标签用于存储节点的标签数据。Neo4j 中的数值、字符串、坐标、日期、时间等属性数据类型分别转换为 ElasticSearch 中的对应数据类型；

（2）**属性写入及更新**：为了保证 Neo4j 数据库中的本地数据和 ElasticSearch 中的数据保持一致，PandaDB 对 Neo4j 中事务操作执行模块（Operations）中的节点更新部分进行了扩展，设计 ExternalPropertyStore 用于存储在 Neo4j 事务中执行的所有操作。例如当 Neo4j 数据库执行插入节点、添加标签、设置属性、删除节点等操作的同时，也将对应的操作数据缓存到 ExternalPropertyStore 中，当 Neo4j 数据库执行事务提交操作的同时将缓存的操作数据同步到 ElasticSearch 中；

（3）**属性过滤**：为了基于协存实现节点属性过滤，PandaDB 对 Neo4j 的 cypher 查询执行计划进行了改造，将节点属性过滤谓词下推到协存管理模块。然后根据谓词过滤条件生成 ElasticSearch 的检索请求，最后将命中的文档（节点）列表返回给查询引擎做进一步筛选。为了避免大量查询结果增大网络传输延迟，PandaDB 采用了异步分批的方式传递的查询结果。

3.5 AI算法调度

AI 算法调度主要包括本地算法驱动管理和 AI 算法服务框架。

（1）**本地算法驱动管理**：为针对不同类型的 BLOB 执行不同的抽取器（SubPropertyExtractor）和语义比较器（SemanticCompartor），制定了如图 11 所示的驱动管理规则库。图 11 中的 DogOrCatClassifier 用以抽取宠物类型，适用于 blob/image 类型的属性；CosineStringSimilarity 用以计算两个文本串的余弦相似度，仅适用于两个 string 类型的属性。

chinaXiv:202007.00035v1

chinaXiv:202007.00035v1

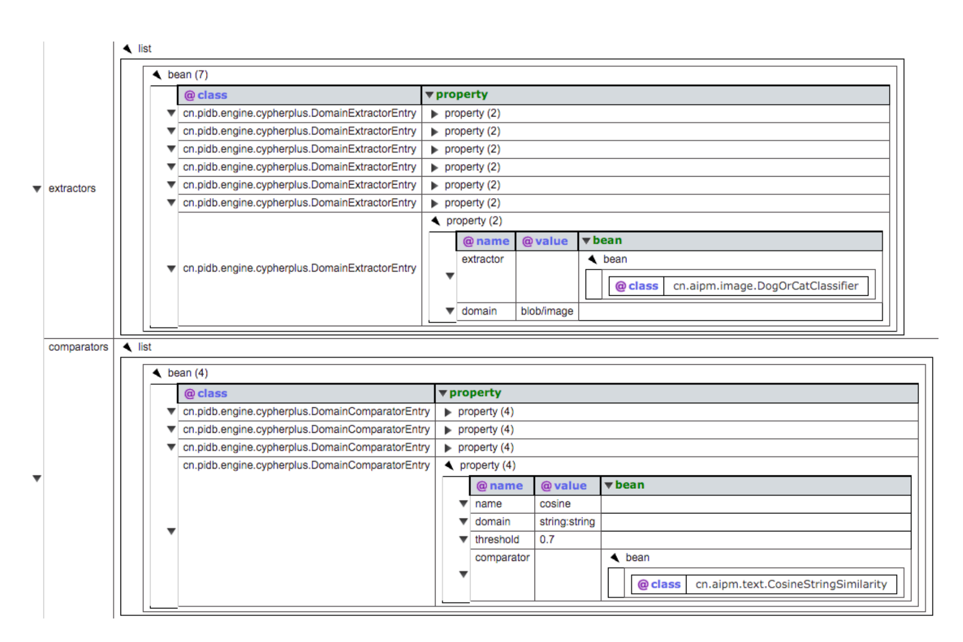


Fig.11 Extractor and Semantic comparator matching rule set

图 11 抽取器和语义比较器匹配规则库

（2）**AI 算法服务框架：**AIPM 用以屏蔽不同 AI 模型之间的依赖冲突问题，降低人工智能模型的部署和维护难度，便于 PandaDB 按需扩展 AI 算子。该模块通过容器技术屏蔽了不同算法之间的依赖冲突，并实现了 AI 算子的快速部署。图 12 给出了 AIPM 与系统之间的交互逻辑。系统以 HTTP 请求的方式向 AIPM 发出查询请求，请求的路径与 AI 算法具有对应关系，AIPM 接受到查询请求，调用对应的 AI 算法处理数据，以 Json 字符串的形式返回结果。为了增强 AI 算子的可扩展性，AIPM 设计了统一的集成接口，并要求算子提供对这些接口的支持。

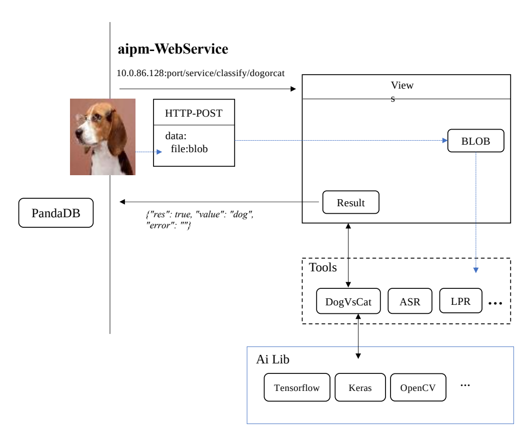


Fig.12 Interaction of PandaDB and AIPM

图 12 AIPM 与 PandaDB 交互框架

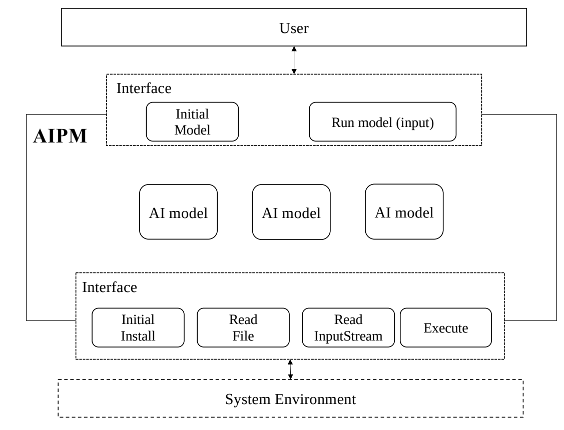


Fig.13 AI model management Framework

图 13 AI 模型管理框架

3.6 分布式架构

为提升对高并发查询请求的响应能力，PandaDB 采用分布式架构。通过 Zookeeper 维护当前可用的节点列表、集群最新数据版本和集群状态。集群共有四种状态，分别是 Ready、PreWrite、Writing、Written，其中

Ready 表示集群正常对外提供服务，响应一切读、写请求；PreWrite 和 Writing 分别是预写状态和写状态，不响应新的请求；Written 表示写完成，立即跳转到 Ready 状态。用户程序通过 PandaDriver 与集群交互，图 14 是 PandaDB 对写请求的响应示意图，当 PandaDriver 接收到用户发出的写请求时，响应步骤如下：

（1）判断当前节点状态，只有当前集群处于 Ready 状态时，才执行下一步操作，否则等待集群变为 Ready 状态。若集群处于 Ready，则通过 Zookeeper 选举 leader node；

（2）建立与 leader node 之间的连接，向 leader node 发出写请求；

（3）leader node 向其他节点发出预写通知，并将集群状态置为 PreWrite，此时不再响应新的读/写请求，对于原有的读请求，继续提供服务直至结束；

（4）当系统中没有正在执行的读任务后，leader node 向其他节点发出写指令，指令的内容为 CypherPlus 语句；

（5）各节点执行 CypherPlus 语句开始写数据，系统进入 Writing 状态。PandaDB 采用属性外置的设计，因此对外置属性的写操作仅由 leader node 发起；

（6）写操作完成后，leader node 向 Zookeeper 更新数据版本，数据版本的作用是，判断节点本地数据是否是最新的，集群状态置为 Written；

（7）返回结果，集群状态变为 Ready。

chinaXiv:202007.00035v1

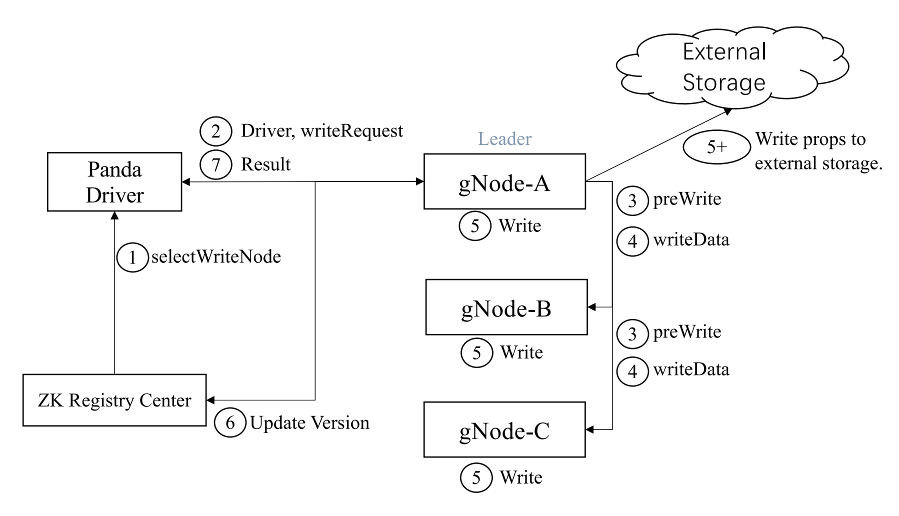


Fig.14 Process of write request handling in PandaDB cluster

图 14 PandaDB 集群写操作处理流程示意图

Table.2 Step and cluster status in write progress

表 2 写操作响应步骤与集群状态

操作序号	集群状态
1,2	Ready
3	PreWrite
4,5	Writing
6	Written
7	Ready

图 15 是系统对读请求的响应流程示意。当用户发出读请求时，PandaDriver 在 Zookeeper 上随机选取一个可用节点，建立与该节点的连接，然后发送查询语句。PandaNode 接收查询语句后，进行语句解析、语义分析、生成逻辑计划、执行等过程，并将结果返回给 PandaDriver。

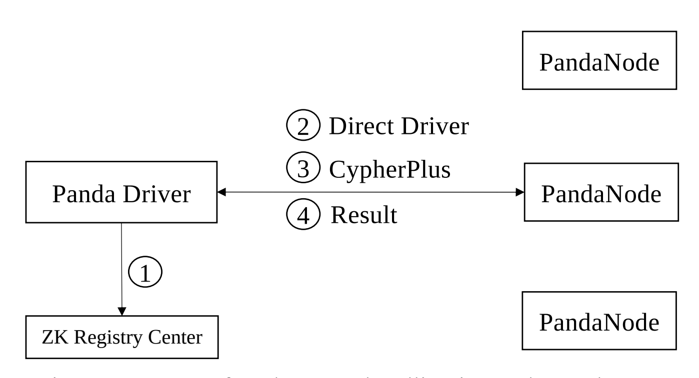


Fig.15 Process of read request handling in PandaDB cluster

图 15 PandaDB 集群读操作处理流程示意图

4 系统效果评估

为充分验证 PandaDB 的性能和功能，本文设计了 5 个测试实验或案例，针对属性协存和分布式方案进行性能测试，同时通过 3 个应用案例验证 PandaDB 对结构化和非结构化数据的融合管理能力。

4.1 属性协存性能测试

为验证基于 ElasticSearch 属性协存方案的性能，设计本测试，对 Neo4j 和引入协存方案后的 PandaDB 的查询性能进行对比，测试环境如表 3 所示。

Table.3 Information about test environment

表 3 测试环境信息

测试环境	环境描述
服务器软硬件环境	5 台同等配置的物理服务器，单台服务器配置为： CPU：32 颗 Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz 内存：128GB 硬盘：6TB 7200 RPM SAS HDD 网络：1000 mbps 互联 操作系统：CentOS Linux release 7.7.1908 (Core)
测试软件版本	Neo4j：3.5.6 （社区版） PandaDB：v0.0.2（开发版） ElasticSearch：6.5.0
环境部署	5 台服务器均部署了 ElasticSearch； 4 台服务器部署 PandaDB； 1 台服务器部署 Neo4j；（注：该 Neo4j 版本仅支持单节点部署）
测试数据	图数据集：1 亿节点（包含 6 种标签），17 亿关系（包含 10 种类型） （注：此数据集来自于实际科技知识图谱项目，其中实体包括人员、机构、论文等内容。）

测试的数据集选取的 Cypher 查询语句如表 4 所示，分别进行多次测试并取执行时间的平均值。为避免冷启动带来的性能影响，测试前分别对系统进行了预热。

Table.4 Query statement in propery co-store test

表 4 协存方案验证测试的查询语句

编号	测试语句	测试类型描述
Q-1	match (n:paper) where n.country="Malta" return count(n) ;	节点查询（单属性精准过滤）
Q-2	match (n:person) where n.org STARTS WITH "Shanghai" return count(n);	节点查询（单属性模糊匹配）
Q-3	match (n:paper) where n.country = "United States" and n.citation = 10 return count(n) ;	节点查询（双属性精准过滤）
Q-4	match (n:person) where n.org STARTS WITH "Shanghai" and n.citations=1 return count(n);	节点查询（模糊匹配与精准匹配结合的双属性过滤）
Q-5	match (n:person) where n.citations=100 and n.citations5=200 return count(n);	节点查询（双属性精准过滤）

chinaXiv:202007.00035v1

Q-6	match (n:person) where n.citations=192 and n.citations5=204 and n.nationality="France" return count(n);	节点查询（三属性精准过滤）
Q-7	match (n:person) where n.citations=192 and n.citations5=204 and n.nationality="France" and n.publications5=5 return count(n);	节点查询（四属性精准过滤）
Q-8	match (startNode:paper)-[r]->(other) where startNode.country = "Japan" and startNode.citation = 5 return count(other);	关系查询（返回末端节点信息）
Q-9	match (n:person)-[r:work_for]->(other) where n.org starts with "Beijing" and n.citations = 1 return count(r)	关系查询（返回关系信息）
Q-10	match (startNode: person { personId:'32'}) optional match (startNode)-[r:write_person] -(other) return count(other)	关系查询（可选关系匹配）

Table.5　Test result for property co-store

表 5　协存方案验证测试结果

测试语句编号	平均执行时间（单位：ms）		执行时间比值
	Neo4j	PandaDB	
Q-1	184	189	0.97
Q-2	96.5	97.5	0.98
Q-3	1001.5	55	18.20
Q-4	426	42	10.14
Q-5	358	309	1.15
Q-6	304.5	51.5	5.91
Q-7	94	52.5	1.79
Q-8	2,658	2,210	1.20
Q-9	453	415	1.09
Q-10	53	62	0.85

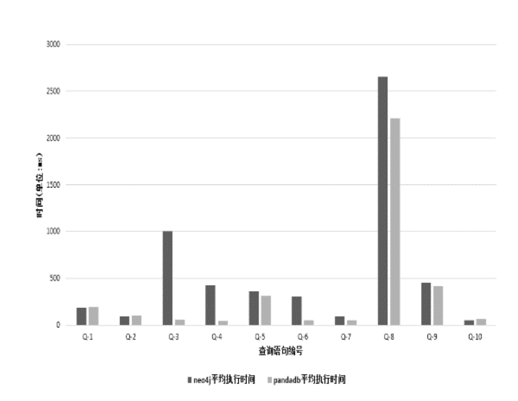


Fig.16　Comparison of query time in co-store

图 16　协存方案查询响应时间对比

从测试结果可看出，由于采用了 ElasticSearch 作为节点属性的协存和索引，在上述查询语句执行测试中，PandaDB 占据明显优势，尤其在节点多属性过滤查询和模糊匹配查询中性能平均提升 2~6 倍。

4.2　分布式性能测试

为验证分布式架构带来的查询相应能力的提升，在独立的物理机集群上部署了 PandaDB，同时部署了 Neo4j 单机版，对分布式查询性能进行评估，测试环境如表 3 所示。

Cypher 语句如表 6 所示，分别进行了冷启动和热启动两组测试，测试中的 Cypher 语句在 Neo4j 和 PandaDB 中均是并发执行的。

Table.6　Query statement in distributed solution test

表 6　分布式方案测试使用的查询语句

编号	Cypher 语句
Q-1	Match (n:paper) where toInteger(n.publishDate)>20000000 return distinct n.paperType, count(n.paperType)
Q-2	Match(n) return distinct labels(n), count(n)
Q-3	Match (n)-[r:org_paper]->(m) where n.cnName contains '医院' AND m.citation>5 Return count(m)
Q-4	Match(n:person) return n.chineseName Order By n.influenceScore Desc limit 25
Q-5	Match(n:person) With distinct n.chineseName as name, count(n.chineseName) as counts where counts>1 return name, counts
Q-6	Match p=(n1:dictionary_ccs) <- [r1:criterion_belong_ccs] - (n2:criterion) <- [r2:org_criterion] - (n:organization) - [r:org_criterion] -> (n3:criterion) - [r3:criterion_belong_ccs] -> (n4:dictionary_ccs) Where not n1.dictionaryId = n4.dictionaryId return count(n)
Q-7	Match(n:paper) where n.paperType contains '期刊' return count(n)
Q-8	Match(n:patent) where toInteger(n.awardDate) > 20180000 return count(n)
Q-9	Match(n:paper_keywords) where n.times_all>1 return count(n)
Q-10	Match(n:patent) where n.chineseName contains '装置' return count(n)

chinaXiv:202007.00035v1

Table.7 Response time for queries in distributed solution test

表 7 分布式方案查询响应时间（单位：ms）

Cypher 语句	冷启动			热启动		
	响应时间（单位：ms）		比值	响应时间（单位：ms）		比值
	Neo4j	PandaDB		Neo4j	PandaDB	
Q-1	超时	262,915	-	89,141	66,487	1.34
Q-2	202,853	68,989	2.94	103,148	58,021	1.77
Q-3	超时	超时	-	295,611	162,833	1.81
Q-4	超时	163,062	-	73,422	45,873	1.60
Q-5	超时	271,975	-	179,365	152,377	1.17
Q-6	156,421	47,506	3.29	81,991	52,755	1.55
Q-7	135,344	113,123	1.19	28,887	16,659	1.73
Q-8	398,781	170,393	2.34	20,733	18,613	1.11
Q-9	478,595	7,786	61.46	8,067	7,864	1.02
Q-10	1,068,889	7,437	143.72	6,364	9,614	0.66

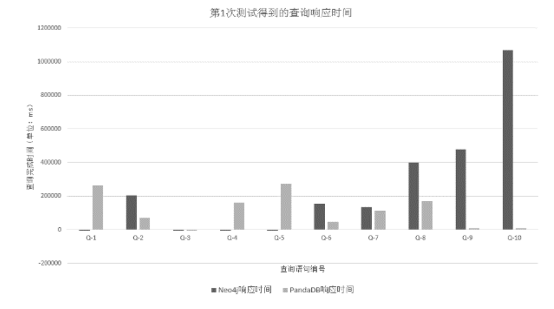


Fig.17 Comparison of response time in cold boot test

图 17 分布式方案冷启动查询响应时间对比

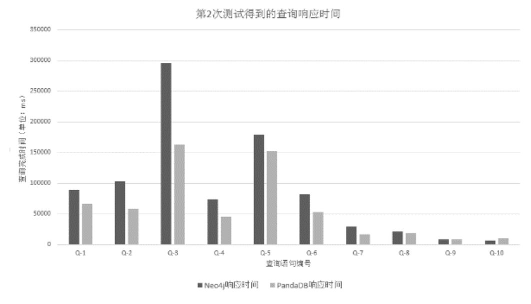


Fig.18 Comparison of response time in hot boot test

图 18 分布式方案热启动查询响应时间对比

chinaXiv:202007.00035v1

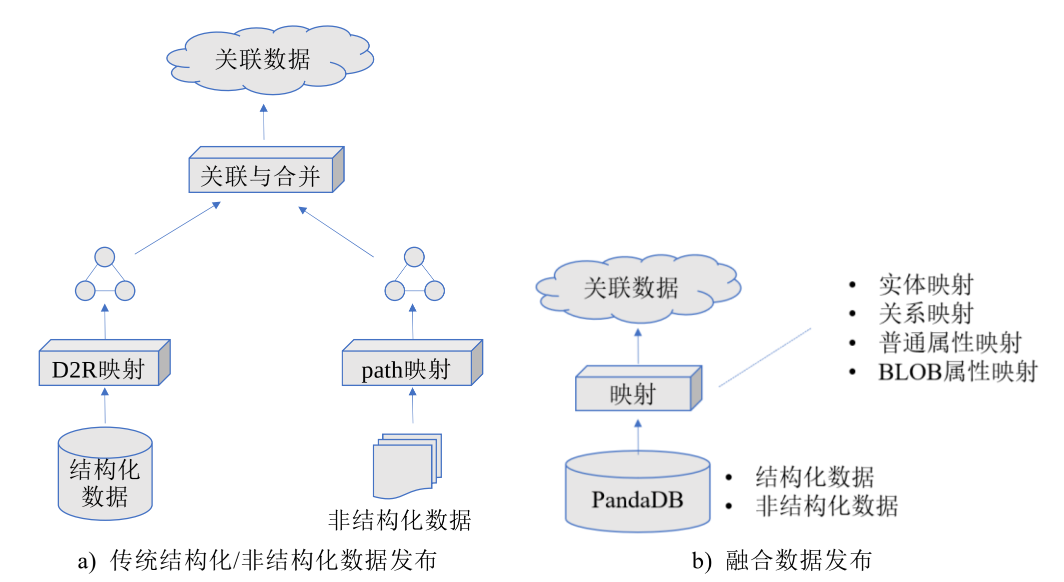
表 7 是在冷启动和热启动条件下分别测试得到的结果，图 17 和图 18 是对结果的对比展示，其中单次查询响应时间超过 300 秒则认为系统无响应。由测试结果可以看出，在上述并发测试中，采用了 4 个节点的分布式架构方案的 PandaDB 平均响应时间是单节点的 Neo4j 的 1/2 到 1/3。

4.3 案例1：关联数据发布

关联数据（Linked Data）的概念由 Berners-Lee 于 2006 年提出[9]，其原理是用一种轻型的、可利用分布数据集及其自主内容格式、基于标准的知识表示与检索协议、可逐步扩展的机制实现可动态关联的知识对象网络，并支持在此基础上的知识组织和知识发现。

关联数据的发布机制和方法研究，即如何把原始数据发布成机器可理解、Web 可访问的 RDF 数据，是关联数据一直以来的研究热点[10]。目前绝大部分的关联数据集都采用两种方式来发布数据：一种诸如 D2RQ[11]等在线映射的方法，即将存储在关系数据库中的数据转换成关联数据；另外一种则采用诸如 Virtuoso、3Store 等系统将处理完的 RDF 数据存储起来，再进行发布。这两种方案存在着明显的不足，即无法很好的处理非结构化数据以及与结构化数据之间的关联。在关联数据中，非结构化数据可以发布成普通的 HTTP URL，但由于数据库本身不具备非结构化数据的管理能力，因此需要额外的手段来发布非结构化数据的内容，并维护它们与结构化信息之间的关联。

如图 19 所示，采用 PandaDB 的融合管理引擎，可以解决关联数据发布过程中结构化、非结构化数据的统一发布问题。



a) 传统结构化/非结构化数据发布 b) 融合数据发布

Fig.19 Changes on ProcessComparison of Linked Publishing Linked Data

图 19 关联数据发布方式与流程变化

本案例借鉴 D2RQ 的映射机制，制定了实体结构化信息和非结构化信息的映射规则：

（1）实体映射：将一条实体映射成一个 RDF Resource，默认 HTTP URI 为 http://<baseuri>/<nodeid>；

（2）标签映射：实体的标签映射成 is_a 语句；

（3）关系映射：实体间的关系映射成 RDF link，链接的谓词采用关系的标签；

（4）属性映射：每个属性映射成一条<主,谓,宾>三元组，属性名称被映射成一个 RDF 谓词；

（5）BLOB 值映射：将 BLOB 值发布成一条 Web 可访问的 HTTP URI，可以定制其路径，默认路径为 http://<baseuri>/<blobid>，将根据元数据信息自动将 Length 和 ContentType 写入 HTTP Response 的头部；

通过在线映射生成的关联数据集效果如图 20 所示。与传统的发布方式相比，该方案支持非结构化数据的发布，同时自动维护与实体之间的关联。

chinaXiv:202007.00035v1

```
<rdf:Description rdf:about="http://zoo.csdb.cn/csdb/resource/entity/12053084">
        <skos:prefLabel>斑头雁</skos:prefLabel>
        <skos:altLabel>白头雁</skos:altLabel>
        <skos:altLabel>黑纹头雁</skos:altLabel>
        <bio:origin>印度及中国西藏</bio:origin>
        <bio:parent rdf:resource="http://zoo.csdb.cn/csdb/resource/entity/120530"/>
        <bio:photo rdf:resource="http://zoo.csdb.cn/csdb/resource/blob/55439"/>
</rdf:Description>
```

http://zoo.csdb.cn/csdb/resource/blob/55439

content-length: 53920
content-type: image/jpeg



Fig.20 Publishing Linked data based on PandaDB

图 20 基于 PandaDB 实现关联数据的发布

4.4 案例2：个人相册管理

照片属于典型的非结构化数据，目前通常采用文件系统进行管理。为了支持扩展信息的检索，往往需要在文件系统之外增加结构化信息的标注信息，如：人物、事件、地点等。另外，如果需要针对照片内容进行高级检索，如：查找所有出现小狗的照片，则需要开发、部署 AI 程序，通过进程调用实现检索目的。可以看出，这种多样化的需求需要借助于 3 个独立的系统完成，如图 21 所示。

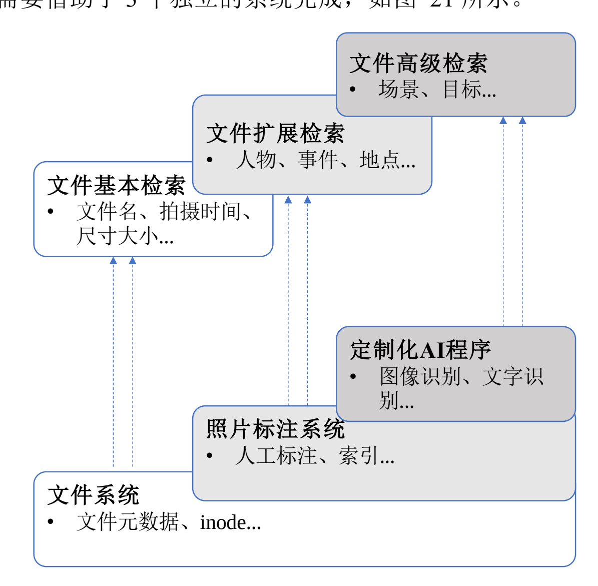


Fig.21 Diversified queries for albums requires building multiple systems

chinaXiv:202007.00035v1

图 21　相册的多样化查询需要构建多个系统

本案例基于 PandaDB 构建个人相册管理原型系统，该系统支持文件系统的图片批量导入，采用智能属性图模型实现图片的管理，同时提供了开放的标注接口，可以为图片增加新的属性。借助于 PandaDB 的次级属性抽取能力，原型系统可支持针对图片的高级搜索，如：查找所有出现小狗的照片，查找所有包含小孩的照片，以及查找所有相似的照片，如图所示。

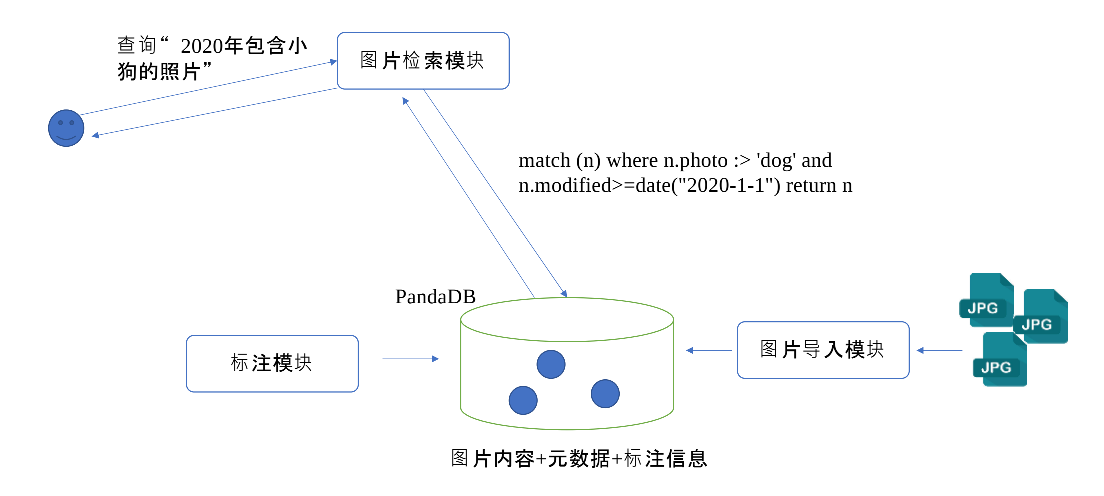


Fig.22　PandaDB supports diversified query requirements

图 22　PandaDB 可满足多样化查询需求

4.5 案例3：学术图谱实体消岐

学术图谱泛指以学术内容为主体的领域知识图谱，AMiner[12]和 AceKG[13]是其中的典型代表。实体消岐的目的是解决信息中名称指向不明及名称歧义问题，学术知识图谱数据中通常出现学者重名、不同机构的简称或缩写相同等情况，因此图谱创建过程中通常面临实体消岐问题。目前大多数实体消岐方法都以聚类为基础，文献[14][15][16] 使用表层特征值计算实体间相似度，但这种方法不能充分利用上下文特征。因为基于表层特征的方法面临信息不足的问题，有学者尝试引入外部信息，如 WikiPedia 中的信息，将这些知识资源作为实体的扩展特征，辅助提升聚类准确性，代表工作如[17][18]。文献[19][20]使用图计算的方法，充分利用网络数据的结构信息，但并不具有很好的普适性。近年来，随着深度学习技术的巨大进步，有学者利用 embedding 技术[21]和神经网络技术[22]这类方法相比于传统方法具有更好的效果，但准确性方面仍不能满足人们的预期。本案例基于 PandaDB，提出了一种融合结构化属性和非结构化属性的消歧方法。如图 23 所示，某学术图谱中有两个姓名分别为 Park Bill 和 Tom Green 的人物节点以及一个名为 Data Vis 的论文节点，需要判断 T.Green 和 Tom Green 是否为同一实体，从而判断 Park Bill 和 Tom Green 之间是否存在合作关系。由于论文中存在的属性数据不足，消歧很有困难。基于 PandaDB 对非结构化信息抽取及语义比较能力，可以充分利用 Tom Green 的照片以及论文作者照片列表，从而达到消歧目的。图 23 下半部分给出了完成该操作的 CypherPlus 语句，其中<:操作符表示包含关系，只需要找到对应的节点，判断 n.photo<:p.screenshot 成立与否，即可计算出二者之间是否存在合作关系。

chinaXiv:202007.00035v1

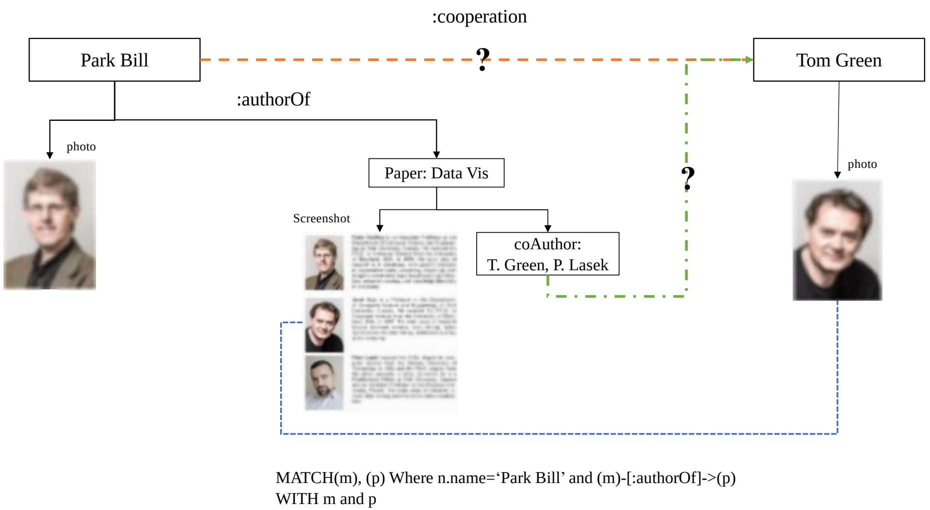


Fig.23 Diagram of Entity Disambiguation based on PandaDB

图 23 基于 PandaDB 实现实体消歧示意图

5 相关工作

AI 与数据库的交叉研究一直是学术界研究的热点内容。AI 与数据库的融合存在 AI4DB（AI for Database）和 DB4AI（Database for AI）两个方向[23]。AI4DB 旨在通过 AI 技术提高 DB 的效率和能力，如自动化数据库参数调优[24]-[26]、基数估计[27]、索引推荐[28]、查询优化[29][30]等。DB4AI，是以数据库为 AI 算法提供数据服务，供算法进行训练和学习。例如，借助数据库的统一 SQL 接口，为用户提供自定义的函数协助构建模型；通过数据库张量计算协助模型训练；通过持久化 AI 模型以重复使用。

人工智能技术的发展，使这项技术被大范围地应用到图像识别、语音识别、机器翻译等领域中。卷积神经网络（Convolutional Neural Network， CNN)在物体识别领域被广泛应用并取得不错的成绩[31][35]。文献[31]提出基于深度学习的 R-CNN(Regions with CNN)方法，该方法将传统物体检测的平均精度由 34. 3% 提升到 66%。在之后的几年，又出现了很多代表性的模型，如 SPP-NET(Spatial Pyramid Pooling Network) [32]、Fast R- CNN[33]、Faster R-CNN[34]和 YOLO(You Only Look Once) [35]。这些模型在 R-CNN 的基础上提高了检测精度并缩短了检测的时间，YOLO 甚至可以对图片进行实时检测。语音识别已经有三十多年的研究历史[36]，从上个世纪八十年代的隐马尔可夫模型[37]，到二十一世纪初的帧级别的深度神经网络模型[38]、CTC 模型[39]，到 2012 年的深度循环神经网络模型[40]，再到 2014 年的注意力机制运用到语音识别[41]，再到 2016 年深度卷积神经网络[42]被用于大规模的语音识别系统。语音识别系统从最初的手动提取特征到如今的端对端的神经网络模型，准确率已经接近 97%。AI 技术对非结构化数据的分析能力，以及其在准确性、效率方面的巨大提升为异构数据的信息抽取和语义计算提供了良好的技术基础。

在数据管理领域，属性图模型[43]是一种常用的管理图数据的数据模型，属性图中的节点和关系都可以被赋予标签和关联任意键值对形式的属性[44]。属性图增加了节点和边的信息,同时又没有改变图的整体结构。目前,属性图模型被图数据库业界广泛采用[45][46],包括著名的图数据库 Neo4j[47]、Titan[48]等。Neo4j 是目前应用较为广泛的一款开源图数据库，其具有从原生图数据存储到可视化插件，再到图数据分析插件的丰富生态。JanusGraph[49]是在 Titan 基础上开发的一种基于属性图的分布式图数据库。JanusGraph 采用存储层和查询引擎分离的设计，可以使用 Cassandra 或 HBase 作为存储层。JanusGraph 通过使用第三方分布式索引库

chinaXiv:202007.00035v1

ElasticSearch、Solr 和 Lucene 实现检索功能。其他的图数据库还包括：Amazon 的 Neptune[50]、微软的 Azure CosmosDB[51]、TigerGraph[52]、OrientDB[53]等。

随着大数据时代的来临，海量数据的存储与计算使得单机服务已经无法满足需求，越来越多的任务需要分布式系统支持。保证分布式系统的可靠性和一致性至关重要。解决这个问题的一个著名算法是由 Lamport 提出的 Paxos 算法[54][55]。此后为了适应不同的工程环境，研究人员在 Paxos 的基础上提出了很多新的算法[56]。比较著名的有 Multi-Paxos[57][58]、Liskov 等人提出的 VR (viewstamped replication)算法[59][60]、雅虎公司设计的 ZAB(Zookeeper’s atomic broadcast)算法[61]、Ongaro 等人提出的 Raft 算法[62]等。

6 总结与展望

本文从结构化/非结构化数据进行融合管理、关联计算和即席查询需求角度出发，分析了目前数据融合管理方面缺少统一表示、不支持交互式查询等难点。在此基础上提出了具备对异构数据实现统一表示能力的智能属性图模型，并提出了针对在线查询和计算的属性操作符与查询语法。在第 3 节，本文提出了基于智能图模型的分布式数据融合管理系统 PandaDB，该系统实现了结构化、非结构化数据的高效存储管理，并提供了灵活的 AI 算子扩展机制，具备对多元异构数据内在信息的即席查询能力。测试实验和案例证明，PandaDB 的协存机制和分布式架构具备较好的性能加速效果，同时 PandaDB 可应用在关联数据发布、个人相册管理、学术图谱实体消歧等多元异构数据融合管理的场景。

目前 PandaDB 还存在着一些不足，如：一方面，AIPM 模块与系统相对独立部署，这种模式降低了系统的耦合性，有利于扩展和维护，但面对大规模的非结构化数据信息查询请求时，模块间信息传输的开销较大。未来应研究更为合理的 AI 功能集成机制，结合多元异构数据即席查询的场景特性设计任务调度方法，提升系统性能。另一方面，从智能属性中抽取内嵌式属性的操作，目前还缺乏有效的缓存和预测机制，造成即席抽取的过程延时较大。PandaDB 将进一步结合应用，进一步提升系统的性能和稳定性，从而提升多元异构数据融合管理的能力。

References:

[1] Gaag A, Kohn A, Lindemann U. Function-based Solution Retrieval and Semantic Search in Mechanical Engineering[C]//Proceedings the 17th International Conference on Engineering Design（ICED 09）. 2009: 147-158.

[2] Buneman P, Davidson S, Fernandez M, et al. Adding structure to unstructured data[C]//International Conference on Database Theory. Springer, Berlin, Heidelberg, 1997: 336-350.

[3] Li W, Lang B. A tetrahedral data model for unstructured database[J]. Scientia Sinica(Informationis), 2010, 40(08): 1039-1053 (in Chinese with English abstract).

[4] Gerber D, Hellmann S, Bühmann L, et al. Real-time RDF extraction from unstructured data streams[C]//International semantic web conference. Springer, Berlin, Heidelberg, 2013: 135-150.

[5] Sears R, Van Ingen C, Gray J. To blob or not to blob: Large object storage in a database or a filesystem?[J]. arXiv preprint cs/0701168, 2007.

[6] Zhu Y, Du N, Tian H, et al. LaUD-MS: an extensible system for unstructured data management[C]//2010 12th International Asia-Pacific Web Conference. IEEE, 2010: 435-440.

[7] Zhang Xiao et al. Managing a large shared bank of data by using Free-Table [C]//Proceedings of the 12th Asia-Pacific Web Conference(APWeb 2010), Busan, Korea, Apr 6-8, 2010: 441 446.

[8] Zhou NN, Zhang X, et al. Design and Implementation of Adaptive Storage Management System in MyBUD [J]. Journal of Frontiers of Computer Science & Technology, 2012,6(08): 673-683 (in Chinese with English abstract).

[9] Berners-Lee T. Design Issues: Linked Data[EB/OL]. [2017- 12-29]. https://www.w3.org/DesignIssues/LinkedData.html.

[10] https://www.w3.org/DesignIssues/LinkedData.html.Linked data : open research problems. Consuming Linked Data Tutorial, World Wide Web Conference 2010[EB/OL]. [2012-10]. http://www.slideshare.net/juansequeda/07-openresearchproblems

chinaXiv:202007.00035v1

[11] Bizer C, Seaborne A: D2RQ-treating non-RDF databases as virtual RDF graphs[C]. In: Proceedings of Proceedings of the 3rd International Semantic Web Conference (ISWC2004). 2004: 26

[12] Tang J. AMiner: Toward understanding big scholar data[C] Proceedings of the ninth ACM international conference on web search and data mining. 2016: 467-467.

[13] Wang R, Yan Y, Wang J, et al. Acekg: A large-scale knowledge graph for academic data mining[C]//Proceedings of the 27th ACM international conference on information and knowledge management. 2018: 1487-1490.

[14] Bagga A, Baldwin B. Entity-Based Cross-Document Core f erencing Using the Vector Space Model[C]//36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1. 1998: 79-85.

[15] Pedersen T, Purandare A, Kulkarni A. Name discrimination by clustering similar contexts[C]//International Conference on Intelligent Text Processing and Computational Linguistics. Springer, Berlin, Heidelberg, 2005: 226-237.

[16] Chen Y, Martin J H. Towards robust unsupervised personal name disambiguation[C]//Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). 2007: 190-198.

[17] Cucerzan S. Large-scale named entity disambiguation based on Wikipedia data[C]//Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL). 2007: 708-716.

[18] Han X, Zhao J. Named entity disambiguation by leveraging wikipedia semantic knowledge[C]//Proceedings of the 18th ACM conference on Information and knowledge management. 2009: 215-224.

[19] Hassell J, Aleman-Meza B, Arpinar I B. Ontology-driven automatic entity disambiguation in unstructured text[C]//International Semantic Web Conference. Springer, Berlin, Heidelberg, 2006: 44-57.

[20] Minkov E, Cohen W W, Ng A Y. Contextual search and name disambiguation in email using graphs[C]//Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. 2006: 27-34.

[21] Zhang B, Al Hasan M. Name disambiguation in anonymized graphs using network embedding[C]//Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017: 1239-1248.

[22] Huang H, Heck L, Ji H. Leveraging deep neural networks and knowledge graphs for entity disambiguation[J]. arXiv preprint arXiv:1504.07678, 2015.

[23] Li GL, Zhou XH. XuanYuan: An AI-native database systems. Ruan Jian Xue Bao/Journal of Software, 2020, 31(3):831−844 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5899.htm

[24] hang J, Liu Y, Zhou K, Li G. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: Proc. of the SIGMOD. 2019.

[25] Li G, Zhou X, Gao B, Li S. Qtune: A query-aware database tuning system with deep reinforcement learning. Proc. of the VLDB Endowment, 2019.

[26] Wang W, Zhang M, Chen G, Jagadish HV, Ooi BC, Tan K. Database meets deep learning: Challenges and opportunities. SIGMOD Record, 2016,45(2):1722.

[27] Kipf A, Kipf T, Radke B, Leis V, Boncz PA, Kemper A. Learned cardinalities: Estimating correlated joins with deep learning. In: Proc. of the CIDR. 2019.

[28] Pedrozo WG, Nievola JC, Ribeiro DC. An adaptive approach for index tuning with learning classifier systems on hybrid storage environments. In: Proc. of the HAIS. 2018. 716729.

[29] Krishnan S, Yang Z, Goldberg K, Hellerstein JM, Stoica I. Learning to optimize join queries with deep reinforcement learning. CoRR, abs/1808.03196, 2018.

[30] Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration. In: Proc. of the 1st Int'l Workshop on Exploiting Artificial Intelligence Techniques for Data Management. 2018. 3:13:4.

[31] GIRSHICK R，DONAHUE J，DARRELL T，et al. Rich feature hi- erarchies for accurate object detection and semantic segmentation [C]/ / Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway，NJ: IEEE，2014: 580 -587 .

chinaXiv:202007.00035v1

[32] HE K，ZHANG X，REN S，et al. Spatial pyramid pooling in deep convolutional networks for visual recognition [C]/ / Proceedings of the 13th European Conference on Computer Vision. Berlin: Springer，2014: 346-361.

[33] GIRSHICK R. Fast R-CNN [C]/ / Proceedings of the 2015 IEEE International Conference on Computer Vision. Piscataway， NJ: IEEE，2015: 1440-1448.

[34] REN S Q，HE K M，GIRSHICK R，et al. Faster R-CNN: towards real-time object detection with region proposal networks [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，2015， PP(99): 1-9.

[35] REDMON J，DIVVALA S，GIRSHICK R，et al. You only look once: unified，real-time object detection [EB / OL]. [2016-01-20]. http: / / ai2-website. s3. amazonaws. com / publications / YOLO. pdf.

[36] Hou YM, Zhou HQ, Wang ZY. Overview of speech recognition based on deep learning[J]. Application Research of Computers, 2017, 34(08): 2241-2246 (in Chinese with English abstract).

[37] Juang B H, Rabiner L R. Hidden Markov models for speech recognition[J]. Technometrics, 1991, 33(3): 251-272.

[38] Graves A , Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural Networks, 2005, 18(5–6):602-610.

[39] Graves A , Santiago Fernández, Gomez F . Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks[C]// International Conference on Machine Learning. ACM, 2006.

[40] Graves A . Sequence Transduction with Recurrent Neural Networks[J]. Computer ence, 2012, 58(3):235-242.

[41] Chorowski J , Bahdanau D , Cho K , et al. End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results[J]. Eprint Arxiv, 2014.

[42] Sercu T , Goel V . Advances in Very Deep Convolutional Neural Networks for LVCSR[C]// Interspeech 2016. 2016.

[43] Ehrig H , Prange U , Taentzer G . Fundamental theory for typed attributed graph transformation.[J]. Lecture Notes in Computer ence, 2004, 3256:161-177.

[44] Robinson I , Webber J , Eifrem E . Graph Databases[M]. O'Reilly Media, Inc. 2013.

[45] Wang X, Zou L, Wang CK, Peng P, Feng ZY. Research on Knowledge Graph Data Management: A Survey[J].Ruan Jian Xue Bao/Journal of Software, 2019,30(07):2139-2174 (in Chinese with English abstract).

[46] R. (2012). A comparison of current graph database models. Paper presented at the 2012 IEEE 28th International Conference on Data Engineering Workshops.

[47] The Neo4j Team. The Neo4j Manual v3.4. 2018. https://neo4j.com/docs/developer-manual/current/

[48] Spmallette. Titan—Distributed graph database. 2018. http://titan.thinkaurelius.com/

[49] JanusGraph Authors. JanusGraph—Distributed graph database. 2018. http://janusgraph.org/

[50] Amazon Web Services, Inc. Amazon Neptune—Fast, reliable graph database build for cloud. 2018. https://aws.amazon.com/neptune/

[51] Microsoft Azure. Microsoft Azure Cosmos DB. 2018. https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

[52] TigerGraph. TigerGraph—The first native parallel graph. 2018. https://www.tigergraph.com/

[53] Callidus Software Inc. OrientDB-multi-model database. 2018. http://orientdb.com/

[54] Lamport L.The part-time parliament[J].ACM Transactions on Computer Systems, 1998, 16 (2) :133-169

[55] Lamport L.Paxos made simple[J].ACM SIGACT News, 2001, 32 (4) :18-25

[56] Wang J, Zhang MX, Wu YW, Chen K, ZhengWM. Paxos-like Consensus Algorithms:A Review[J]. Journal of Computer Research and Development, 2019, 56(04): 692-707 (in Chinese with English abstract).

[57] Lamport L.Paxos made simple[J].ACM SIGACT News, 2001, 32 (4) :18-25

[58] Chandra T D, Griesemer D, Redstone J.Paxos made live:An engineering perspective[C]//Proc of the 26th Annual ACM Symp on Principles of Distributed Computing (PODC’07) .New York:ACM, 2007:398-407

[59] Oki B, Liskov B.Viewstamped replication:A general primary-copy method to support highly-available distributed systems[C]//Proc of the 7th Annual ACM Symp on Principles of Distributed Computing (PODC’88) .New York:ACM, 1988:8-17

[60] Liskov B, Cowling J.Viewstamped replication revisited[R].Cambridge, MA:MIT CSAIL, 2012

[61] Medeiros A.Zookeeper’s atomic broadcast protocol:Theory and practice[R].Otakaari, Espoo:Aalto University, School of Science, 2012

chinaXiv:202007.00035v1

[62] Ongaro D, Ousterhout J.In search of an understandable consensus algorithm[C]//Proc of the 2014 USENIXAnnual Technical Conf (ATC’14) .Berkeley, CA:USENIXAssociation, 2014:305-319

附中文参考文献:

[3] 李未，郎波. 一种非结构化数据库的四面体数据模型[J]. 中国科学：信息科学, 2010, 40(08): 1039-1053.

[8] 周宁南,张孝,孙新云,琚星星,刘奎呈,杜小勇,王珊.MyBUD 自适应分布式存储管理的设计与实现[J].计算机科学与探索,2012,6(08):673-683.

[23] 李国良,周煊赫.轩辕:AI 原生数据库系统[J].软件学报,2020,31(03):831-844.

[36] 侯一民,周慧琼,王政一.深度学习在语音识别中的研究进展综述[J].计算机应用研究,2017,34(08):2241-2246.

[45] 王鑫,邹磊,王朝坤,彭鹏,冯志勇.知识图谱数据管理研究综述[J].软件学报,2019,30(07):2139-2174.

[56] 王江,章明星,武永卫,陈康,郑纬民.类 Paxos 共识算法研究进展[J].计算机研究与发展,2019,56(04):692-707.

chinaXiv:202007.00035v1